# GERADOR AUTOMÁTICO DE PLANOS DE ENSINO: EXTRAÇÃO DE DADOS EM PDF E GERAÇÃO DE DOCUMENTOS COM PYTHON

Matheus José Felipe Rosa[1], Daniel Almeida Colombo[2] e Willian Candido Pedroso[3]

## RESUMO

O presente trabalho apresenta o desenvolvimento de uma aplicação web destinada à automação da criação de Planos de Aula e Ensino (PPDs) para o Centro Universitário Senai Paraná (UniSenai-PR). O objetivo principal foi solucionar a morosidade e a suscetibilidade a erros inerentes ao processo manual de transcrição de dados contidos nos Projetos Pedagógicos de Curso (PPCs) em formato PDF para os templates institucionais em Microsoft Word. A metodologia adotada envolveu a construção de uma arquitetura modular em Python, utilizando a biblioteca PyMuPDF para a extração estruturada de textos via expressões regulares (regex), a biblioteca DocxTemplate para a injeção automatizada de dados no documento final, e o framework Streamlit para a criação de uma interface de usuário acessível. Os resultados indicam que a ferramenta reduz o tempo de confecção dos documentos de horas para segundos, garantindo a padronização e a formatação exigidas pelas normas da instituição (ABNT). Conclui-se que o sistema não apenas otimiza o fluxo de trabalho docente e administrativo, mas também serve como base escalável para futuras automações no ambiente acadêmico.

**Palavras-chave:** Automação. Python. Extração de dados. Streamlit. Documentos educacionais.

## AUTOMATIC TEACHING PLAN GENERATOR: PDF DATA EXTRACTION AND DOCUMENT GENERATION WITH PYTHON

## ABSTRACT

This paper presents the development of a web application aimed at automating the creation of Teaching and Lesson Plans (PPDs) for the Centro Universitário Senai Paraná (UniSenai-PR). The main objective was to solve the slowness and error susceptibility inherent in the manual process of transcribing data contained in Course Pedagogical Projects (PPCs) in PDF format to institutional templates in Microsoft Word. The methodology adopted involved building a modular Python architecture, using the PyMuPDF library for structured text extraction via regular expressions (Regex), the DocxTemplate library for automated data injection into the final document, and the Streamlit framework for creating an accessible user interface. The results indicate that the tool reduces document creation time from hours to seconds, ensuring the standardization and formatting required by institutional norms (ABNT). It is concluded that the system not only optimizes the teaching and administrative

[1] UniSenaiPR Londrina - E-mail: pwvmat@gmail.com
[2] UniSenaiPR Londrina - E-mail: daniel.colombo@sistemafiep.org.br
[3] UniSenaiPR Londrina - E-mail: willian.pedroso@sistemafiep.org.br

workflow but also serves as a scalable basis for future automations in the academic environment.
**Keywords:** Automation. Python. Data extraction. Streamlit. Educational documents.

## 1. IDENTIFICAÇÃO DA EMPRESA

Centro Universitário SENAI Paraná - Cód. MEC 1400

CNPJ: 03.776.284/0001-09.

Razão Social: Serviço Nacional de Aprendizagem Industrial - Senai.

Londrina – PR.

Atividade Econômica Principal: Outras atividades de ensino não especificadas anteriormente.

Código CNAE Principal: 85.99-6-99 (Outras atividades de ensino não especificadas anteriormente).

Porte: Sem Enquadramento.

## 2. INTRODUÇÃO

A gestão acadêmica envolve uma série de processos burocráticos e documentais essenciais para a manutenção da qualidade do ensino. Entre esses processos, destaca-se a elaboração de Planos de Aula e Ensino (PPDs), documentos que orientam as atividades letivas com base nas diretrizes estabelecidas pelos Projetos Pedagógicos de Curso (PPCs). Historicamente, a criação desses planos exige que os docentes ou a equipe pedagógica extraiam manualmente informações de PDFs extensos (PPCs) e as transcrevam para templates institucionais, um esforço operacional redundante e suscetível a falhas humanas (Sweigart, 2019).

## 3. OBJETIVO GERAL

Este trabalho de Iniciação Científica — desenvolvido como contrapartida e equivalente ao estágio supervisionado em Engenharia de Software — tem como objetivo geral desenvolver uma aplicação web capaz de automatizar a extração de dados de ementas em PDF e gerar os respectivos documentos em formato Word (.docx).

## 4. OBJETIVO ESPECÍFICO

Os objetivos específicos incluem: criar um motor de extração de texto baseado em expressões regulares (Regex); desenvolver um injetor de dados para templates Word que respeite a formatação ABNT; e construir uma interface gráfica intuitiva para usuários sem conhecimento em programação.

A modernização desse fluxo de trabalho não apenas atende a uma demanda interna de eficiência da UniSenai-PR, mas também demonstra a aplicação prática de conceitos de Engenharia de Software na resolução de problemas administrativos.

## 5. FUNDAMENTAÇÃO TEÓRICA

O desenvolvimento de ferramentas voltadas à automação de processos acadêmicos e administrativos baseia-se em três pilares conceituais da Engenharia de Software: a extração e o *parsing* de fluxos de dados textuais a partir de documentos digitais, o uso de linguagens regulares para filtragem determinística e o padrão arquitetural de divisão de responsabilidades.

### 5.1 Desestruturação Semântica em Arquivos PDF

O formato PDF (Portable Document Format), embora seja o padrão global para intercâmbio de documentos finais devido à sua fidelidade visual independente de plataforma, apresenta severas limitações para a reutilização de dados (Nada; Linden, 2020). Concebido originalmente para representar páginas de forma gráfica e estática, o arquivo PDF armazena informações textuais através de coordenadas bidimensionais (X, Y) em um plano de renderização.

Como consequência dessa especificação, conceitos semânticos estruturais — tais como parágrafos, tabelas, quebras de linha lógicas e fluxos contínuos de texto — são completamente perdidos na compilação do arquivo. O texto torna-se um agregado de caracteres ou strings soltas posicionadas no espaço.

Ao realizar a extração bruta de dados através de bibliotecas de manipulação de streams de PDF, como o PyMuPDF, o resultado frequentemente apresenta ruídos estruturais, tais como hífens de quebra de linha órfãos, quebras de linha arbitrárias

geradas por formatação de colunas e inclusão indesejada de elementos repetitivos de layout (como cabeçalhos e rodapés institucionais). Portanto, a recuperação da informação útil exige uma etapa subsequente de normalização de strings e processamento de texto.

### 5.2 Expressões Regulares (Regex) como Mecanismo de Parsing Determinístico

Para processar os dados textuais extraídos e isolar blocos específicos de informação (como ementas, competências e bibliografias), utiliza-se o conceito de Expressões Regulares (Regular Expressions ou Regex). Matematicamente, uma expressão regular é uma sintaxe que define uma linguagem regular, o nível mais simples da hierarquia de linguagens formais de Chomsky (Sipser, 2012).

As linguagens regulares são computacionalmente processadas por Autômatos Finitos Determinísticos (AFD). Na prática do desenvolvimento de software, isso significa que um motor de Regex consegue varrer uma cadeia de caracteres em tempo linear em relação ao tamanho do texto, buscando correspondências a padrões específicos sem a necessidade de processamento semântico complexo.

De acordo com Friedl (2006), o uso de metacaracteres e quantificadores permite a criação de filtros capazes de identificar âncoras textuais fixas — tais como delimitadores de seção em letras maiúsculas — e capturar o conteúdo dinâmico situado entre eles. No cenário de extração de matrizes curriculares, as expressões regulares atuam como um parser determinístico ideal, dado que a estrutura dos Projetos Pedagógicos de Curso (PPCs), embora textualmente poluída, segue um padrão fixo e repetível de termos-chave.

### 5.3 Separação de Preocupações (Separation of Concerns) e Modularização

A arquitetura de software adotada no projeto fundamenta-se no princípio de Separação de Preocupações (Separation of Concerns - SoC), um dos conceitos

basilares introduzidos por Edsger Dijkstra e amplamente defendido na engenharia moderna por Robert C. Martin (2019). O princípio estabelece que um sistema deve ser dividido em seções distintas, onde cada uma aborda uma responsabilidade ou "preocupação" específica do software.

No contexto da aplicação desenvolvida, a arquitetura foi mapeada em três módulos independentes:

- Camada de Apresentação/Interface (Streamlit): Responsável exclusiva pela interação com o usuário, captura de arquivos e exibição de feedbacks textuais, agindo de forma agnóstica em relação às regras de extração de dados.

- Camada de Lógica de Extração (Processamento de Negócio): Onde reside o motor de busca baseado em regex e manipulação de PDFs, responsável por transformar o fluxo textual bruto em estruturas de dados nativas (dicionários).

- Camada de Persistência e Geração de Documentos (DocxTemplate): Responsável por mapear o dicionário de dados gerado e injetá-lo nas tags do template do Microsoft Word, sem qualquer dependência da origem desses dados.

Essa divisão arquitetural garante o baixo acoplamento e a alta coesão do código (Sommerville, 2011). Caso a instituição decida alterar o formato de entrada de PDF para uma API baseada em JSON, ou o formato de saída de Word para PDF direto, apenas o módulo específico precisará ser refatorado, mantendo a integridade do restante do ecossistema de software.

## 6. METODOLOGIA

### 6.1 Extração de Dados e Tratamento Textual

O núcleo do processamento de leitura utiliza a biblioteca PyMuPDF (também conhecida como fitz), que permite a conversão de páginas de arquivos PDF em strings de texto. Como os documentos institucionais possuem formatação engessada,

o texto extraído frequentemente contém quebras de linha indesejadas, rodapés e cabeçalhos repetitivos (ex: "Uni SENAI PR").

Para contornar esses ruídos estruturais nos dados, utilizou-se o módulo re do Python. Foram desenvolvidas expressões regulares cirúrgicas para localizar e extrair blocos de informação específicos, como "EMENTA", "CONTEÚDOS FORMATIVOS", "BIBLIOGRAFIA BÁSICA" e "PERFIL DE SAÍDA". As Regex também foram cruciais para a separação de URLs das referências bibliográficas, formatando-as automaticamente para o padrão exigido na saída.

### 6.2 Geração Automática de Documentos

Em vez de formatar um documento do zero via código, o projeto empregou a biblioteca docxtpl. Essa ferramenta permite o uso de um arquivo Microsoft Word (.docx) existente como um "molde" (template), contendo tags Jinja2 (ex: {{ nome_disciplina }}). O script recebe um dicionário de dados oriundo do extrator de PDF e injeta as informações diretamente nas marcações, preservando fontes, margens, tabelas e estilos originais do template oficial da UniSenai-PR.

### 6.3 Interface do Usuário

Para que a ferramenta pudesse ser operada por qualquer coordenador ou professor, a interface foi desenvolvida utilizando o framework Streamlit. O Streamlit permitiu a criação rápida de uma aplicação web interativa onde o usuário pode realizar o upload da ementa em PDF, preencher campos complementares (nome do professor, coordenador, curso) e baixar o arquivo .docx gerado com um único clique (Streamlit, 2026).

## 7. APRESENTAÇÃO E DISCUSSÃO DOS RESULTADOS

O protótipo final ("AutoPPD") demonstrou eficácia no cumprimento de seus objetivos. A ferramenta foi testada com ementas reais (como "Ementas GRB0000605.pdf") e conseguiu estruturar com sucesso os dados de bibliografia, ementa e competências. O tempo de confecção de um Plano de Ensino foi reduzido

de forma drástica, eliminando o "copiar e colar" manual e mitigando erros de formatação ABNT nas referências.

Do ponto de vista da formação profissional, o projeto proporcionou aprendizados cruciais equivalentes às expectativas de um estágio em desenvolvimento de software:

- Competências Técnicas (Hard Skills): Aprofundamento no ecossistema Python; domínio prático de Expressões Regulares (Regex) para data parsing; manipulação de sistemas de arquivos e I/O de documentos; e implementação de interfaces low-code com Streamlit.
- Visão Arquitetural: Aplicação prática de modularização, separando a lógica de negócios da interface gráfica.
- Resolução de Problemas: Adaptação da lógica para tratar exceções, como a limpeza de textos residuais (rodapés e artefatos de PDF) que poderiam quebrar a formatação do Word.

## 8. CONSIDERAÇÕES FINAIS

O desenvolvimento do Gerador Automático de PPD cumpriu a proposta de automatizar um processo moroso dentro da rotina acadêmica da UniSenai-PR. A integração entre PyMuPDF, docxtpl e Streamlit provou ser uma stack tecnológica leve, altamente eficiente e de fácil manutenção para o contexto proposto.

A experiência obtida ao longo desta Iniciação Científica substituiu de forma prática e aplicada as exigências de um estágio supervisionado, garantindo o desenvolvimento de competências centrais da Engenharia de Software, desde o levantamento do problema real até o deploy da solução.

Como sugestões para estudos e desenvolvimentos futuros, o repositório da aplicação (README.md) já prevê a extração e o autopreenchimento de áreas mais complexas do PPD, como o cronograma diário de aulas, Atividades Práticas Supervisionadas (APS) e composição de nota. Expandir o motor de Regex para

capturar e calcular esses dados elevará o sistema a um novo patamar de automação institucional.

## REFERÊNCIAS


ARTIFEX SOFTWARE. **PyMuPDF Documentation**. 2024. Disponível em: https://pymupdf.readthedocs.io/. Acesso em: 06 jun. 2026.

FRIEDL, Jeffrey E. F. **Mastering Regular Expressions**. 3. ed. Sebastopol: O'Reilly Media, 2006.

MARTIN, Robert C. **Clean Architecture: A Craftsman's Guide to Software Structure and Design**. Boston: Prentice Hall, 2019.

NADA, Aly; LINDEN, Peter V. D. **PDF Reference: Document Management - Portable Document Format**. 2. ed. New York: Adobe Systems, 2020.

PYTHON SOFTWARE FOUNDATION. **Python 3 Documentation: The Python Standard Library (re - Regular expression operations)**. 2026. Disponível em: https://docs.python.org/3/library/re.html. Acesso em: 06 jun. 2026.

SIPSER, Michael. **Introduction to the Theory of Computation**. 3. ed. Boston: Cengage Learning, 2012.

SOMMERVILLE, Ian. **Engenharia de Software**. 9. ed. São Paulo: Pearson Prentice Hall, 2011.

STREAMLIT INC. **Streamlit Documentation**. 2026. Disponível em: https://docs.streamlit.io/. Acesso em: 06 jun. 2026.

SWEIGART, Al. **Automate the Boring Stuff with Python: Practical Programming for Total Beginners**. 2. ed. San Francisco: No Starch Press, 2019.




[Recebido/Received: Dezembro 18 2024; Aceito/Accepted: Janeiro 29, 2025]